

FIG. 2

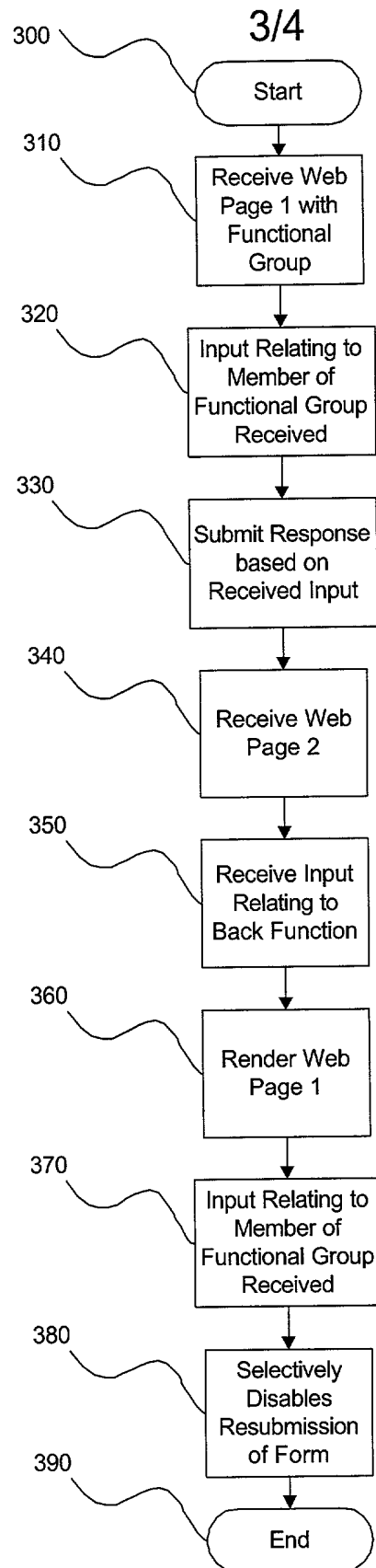


FIG. 3

```

/*=====
PageHistory methods

Description:
5      definition of the PageHistory methods

===== */
function PageHistory_setPageFlag(name, flag) {
10      //variable used in the cookie: literal string with ' to escape the potential : inside of name
      var namelit = "" + name + "";
      //set it in the cookie string
      //case in it is in pages already, then modify value in cookie
      if (!this.isNotInHistory(name)) {
15          var repl = new RegExp(namelit + ":" + "[^\\s]*\\s");
          debug(GenDebug, "this.cookie before", this.cookie);
          debug(GenDebug, "repl", repl);
          this.cookie = this.cookie.replace(repl, namelit + ":" + flag + " ");
          debug(GenDebug, "this.cookie after", this.cookie);
      }
20      else {
          //case cookie empty, create it
          if (this.cookie == "") {
              this.cookie = "{ " + namelit + ":" + flag + " }";
          }
          //else add it to the beginning
          else {
              this.cookie = "{ " + namelit + ":" + flag + " , " + this.cookie.substr(1);
          }
          this.size++;
          debug(GenDebug, "this.size", this.size);
          //case size > maxsize, remove lru = end
          if (this.size > this.maxsize) {
30              debug(GenDebug, "this.cookie before size restraint", this.cookie);
              this.cookie = this.cookie.slice(0, this.cookie.lastIndexOf(",") + 1);
              debug(GenDebug, "this.cookie after size restraint", this.cookie);
          }
      }
      debug(GenDebug, "this.cookie end", this.cookie);
      //set it in the hashtable
40      this.pages[name] = flag;
      //set the cookie
      setCookie("pageh", this.cookie);
  }

45  function PageHistory_getPageFlag(name) {
      if (IsDef(this.pages[name])) return this.pages[name];
  }
  function PageHistory_isNotInHistory(name) {
      if (IsUndef(this.pages[name])) return true;
50      return false;
  }

/*=====
PageHistory()

55      Description:

```

FIG. 4

definition of the PageHistory constructor

The PageHistory object stores a mru/lru (most and least recently used) list of size
maxSize pages for which you submitted a form

When you add a page and the list is full we discard lru and add it as mru

Pages are characterized by a guid and a a flag

The flag indicates if the page has already been submitted

This object has some methods that lets you add a page, update a flag for a page
and get a flag from a page.

```
===== */
//no need to use prototypes since we use only one instance of the object
function PageHistory(maxsize) {
    //define the member variables
    this.maxsize = maxsize;
    this.size = 0;
    //we store a js expression defining an associative array in the cookie
    //we instantiate the array for quick lookups in the js object pages
    //we keep the cookie to manage the lru/mru for serialization
    //this makes the whole thing much faster: remove beginning and add to the end are very easy ops
    on strings
    //and we don't have to loop to serialize/deserialize
    this.cookie = getCookie("pageh");
    debug(GenDebug, "this.cookie", this.cookie);
    if (this.cookie == "") {
        this.pages = new Object();
    }
    else {
        this.pages = eval("bozo = " + this.cookie);
        this.size = this.cookie.split(',').length;
        debug(GenDebug, "this.size", this.size);
    }
    //define the methods
    this.setPageFlag = PageHistory_setPageFlag;
    this.getPageFlag = PageHistory_getPageFlag;
    this.isNotInHistory = PageHistory_isNotInHistory;
    debug(GenDebug, "pages", dumpObject(this.pages));
}

var pageHistory = new PageHistory(20);
```

FIG. 4

```

/*=====
    allowSubmit(name)

    Description:
5      checks if a page is allowed to be submitted. If it is allowed to do so, returns true and sets its
      flag to 0 so that it won't be allowed again.
      to be called on a onSubmit handler

===== */
10  function allowSubmit(name, confirmFlag) {
      uKey = pageHistory.getPageFlag(name)
      debug(GenDebug, "uKey", uKey);
      if (uKey == "1") {
15          pageHistory.setPageFlag(name, "0");
          return true;
      } else {
          if (confirmFlag) {
              return confirm("This form has previously been submitted. Submitting again may result in an
20  error. Do you want to submit?");
          } else {
              alert("This Form has previously been submitted. It cannot be submitted again.");
              return false;
          }
      }
25 }

/*=====
    putPageInHistory(name)

    Description:
30      puts a page in history if it is not there, with a flag allowing it to be submitted.
      To be called at the beginning of your page

===== */
35  function putPageInHistory(name) {
      debug(GenDebug, "pageHistory.isNotInHistory(name)", pageHistory.isNotInHistory(name));
      if (pageHistory.isNotInHistory(name)) {
          pageHistory.setPageFlag(name, "1");
      }
40  }

function dumpObject(obj) {
    var dump = "";
    for (var i in obj) dump += i + "=" + obj[i] + "\n";
45  return dump;
}

```

FIG. 4

```

/*=====
  getObject()

  Description: convert object name string or object reference into a valid object reference
  for both browsers: this is a reference on which you can set some style attributes
===== */
5
function getObject(obj) {
  var theObj;
  if (typeof obj == "string") {
10
    var iniObj;
    if (isNav6) {
      iniObj = document.getElementById(obj);
    }
    else {
15
      iniObj = eval("document." + coll + obj);
    }

    if (isUndef(iniObj)) {
20
      return "undefined";
    }

    if (isNav4) {
      return iniObj;
    }
    else {
25
      // in the IE or NS6 case the iniObj.style object may be undefined
      if (isDef(iniObj.style)) {
        return iniObj.style;
      }
      else {
30
        return "undefined";
      }
    }
  }
  else {
35
    theObj = obj;
  }
  return theObj;
40
}

```

FIG. 4

```

/*=====
  getObjectRef()

  Description: convert object name string or object reference into a valid object reference
  for both browsers, without the style in IE: this is the real object reference
  this function is adapted from Danny Goodman's "Dynamic Html : The Definitive Reference"
  http://www.amazon.com/exec/obidos/ASIN/1565924940/qid%3D963012863/002-0174003-
  8509633

===== */
function getObjectRef(obj) {alert("getRef "+obj);

    var theObj;
    if (typeof obj == "string") {
        var iniObj = eval("document." + coll + obj);
        //alert("getRef "+iniObj);
        if (IsUndef(iniObj)) {
            return "undefined1";
        }
        return iniObj;

    }
    else {
        theObj = obj;
    }
    return theObj;
}

```

FIG. 4


```

/* =====
FUNCTION:    IsUndef

INPUT:      val - the value to be tested

5 RETURN:    true, if the value is undefined
           false, otherwise.

PLATFORMS:  Netscape Navigator 3.01 and higher,
10           Microsoft Internet Explorer 3.02 and higher,
           Netscape Enterprise Server 3.0,
           Microsoft IIS/ASP 3.0.
===== */

15 function IsUndef( val ) {
    var isValid = false;
    if (val+"" == "undefined")
        isValid = true;

    return isValid;
20 } // end IsUndef

function IsDef( val ) {
    return !IsUndef(val);
} // end IsUndef

25 /**
 * <pre>
 * This function
 *     checks if the form is allowed to be submitted.
 *     if yes, it sets the action, then calls form.submit()
 *
 * Usage:
 *
 * <code>
 *     <a href="Javascript:submitForm('/iMM/somefile.jsp', 'myForm', 'someUniqueKey', true)">
 * </code>
 *
 * </pre>
 */
35
40 function submitForm(action, formName, key, confirmFlag) {
    //we put a provision here to let the users of this function not provide the last argument, which the
    //defaults to true
    if (IsUndef(confirmFlag)) {
        confirmFlag = true;
45     }
    if (allowSubmit(key, confirmFlag)) {
        debug(GenDebug, "formName", formName);
        eval("document." + formName + ".action=\"\" + action + \"\"");
        eval("document." + formName + ".submit()");
50     }
    }
}

```

FIG. 4